Tradeoffs in Secure System Development: An Outline

Catherine Meadows
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC 20375

Abstract

In this paper we identify several areas in which the satisfaction of security requirements can affect the cost and performance of a system, and describe what is known about tradeoffs in these areas. We also show where these tradeoffs appear in the life cycle of a system, and show how they are affected by different kinds of security requirements.

1 Introduction

When designing a system of any magnitude, it is necessary to keep in mind that it must satisfy a large number of requirements, some of which may be in conflict with each other. In some cases it may be necessary to trade off the different requirements against each other; in other cases it may be possible to identify other system features that may be traded off to resolve the conflicts between the two requirements. This is as true of security as it is of any other system requirement.

Tradeoffs in security can be complicated by the fact that a system may have a number of different security requirements that lead to demands that are themselves in conflict with each other. Thus the different types of demands that different types of security requirements may put upon a system must be understood. Also, security requirements, as is true of other requirements, can lead to to tradeoffs that affect the system at different points of its life cycle. Thus, it is important to be aware of the system's entire life cycle when identifying tradeoffs.

In this paper we give an overview of the tradeoffs involved in designing secure systems. We do so from three points of view. The first is the different types of security requirements a system may have, broadly organized into confidentiality, integrity, guarantee of service, and authentication. The second is the different points of in a life cycle of a system. The third is four major areas that we have identified as being the most likely places where tradeoffs will occur. These tradeoff areas are: restrictions on communications, required communications, security management overhead, and a system's interaction with its environment.

The rest of this paper is organized as follows. In Section 2, we give a brief taxonomy of system security requirements. In Section 3, we give an outline of the life cycle of a secure system and describe the sorts of requirements that arise at each point. In Section 4, we describe four major tradeoff areas and disuss the tradeoffs that can occur in these areas. We also discuss the way these tradeoffs affect the different types of security requirements, and how they affect a system at each point of its life cycle. In Section 5 we present our conclusions.

2 A Taxonomy of System Security Requirements

Confidentiality

By confidentiality we mean the requirement that a computer system prevent unauthorized access to information. Confidentiality requirements can cover a broad range, from protecting personal data to separating data classified at different security levels.

Integrity

By integrity we mean the prevention of unauthorized modification of data. Integrity can be thought of as a dual to confidentiality.

Prevention of Denial of Service

By prevention of denial of service, we mean the ability to keep hostile intruders from preventing a system from performing its functions. Denial of service can be interpreted in the broad sense as anyway an intruder can do to prevent a system from performing its functions, but it has also been interpreted in the narrower sense (for example in [YG90, Mil94] of an intruder's preventing a system from performing its functions by monopolizing resources. The denial-of-service attack in the secure readers-writers problem discussed above is a classic example of such an attack.

Authentication

By authentication, we mean the ability of an entity to identify correctly an entity it is communicating with, and to identify itself to other entities. Authentication has not traditionally been thought to be a separate requirement on the same level as the other three, but, as we move more an more to large, distributed systems that must communicate through an untrusted medium, it is gaining in importance.

Interactions Between Requirements

Each of the four types of requirements can be used in support of the other. For example, a system that enforces confidentiality may rely on access control lists to determine who is authorized to access what data. These access control lists must be protected from unauthorized modification, thus giving rise to an integrity requirement. As another example, authentication in a network usually requires the ability to demonstrate knowledge of a secret key. The secrecy of this key must be protected, thus giving rise to a confidentiality requirement.

The different types of requirements can also be in conflict with each other. For example, consider the secure readers-writers problem. This problem arises in multilevel secure systems in which a security kernel manages untrusted labeled subjects which are only allowed to read data at their level or below. An untrusted subject cannot write data below its level, since it is not trusted not to write sensitive information. Moverover, the ability of an untrusted subject to affect portions of the system visible to subjects at a low level in restricted as much as possible, since such an ability could be used as a covert channel by which Trojan Horse code in the high subject could signal sensitive information to a low subject.

Now, consider the case in which a high subject is trying to read a low object and low subject is trying to write it. Suppose that the high subject attempts to put a read lock on the object. If this is visible to the low subject, then the high subject might be able to exploit this as a covert channel. Thus, one solution to the problem for the high subject to relinquish its read lock whenever the low

subject wishes to write. This makes the high subject vulnerable to a denial-of-service attack, since the low subject can prevent it from reading the low object by continually requesting to write it.

3 Requirements in a Secure System's Life Cycle

In this section we give the main points of a secure system's life cycle, and discuss the requirements that are relevant to each point. By this we mean, not the requirements that are considered at each point, but the requirements that affect the ability of that portion of the life cycle to be carried out.

System Design and Implementation

What is the cost of building the system? How long does it take to design and build? Does it take any kind of special expertise that may be hard to find?

System Evaluation and Certification

System evaluation and certification can add a great deal to the amount of time it takes to develop a secure system, and can greatly affect its cost. Thus, we also need to ask, not how long it takes to develop a secure system, but how long it takes to evaluate and certify it. Does it clearly fit within accepted security standards? Does it use techniques that have been used in other successfully evaluated systems, or does it introduce new ones that may take longer to gain acceptance? Is the argument for system security understandable or convoluted?

System Use

In this area we put the requirements that a system will have to satisfy when it is actually being used. These include all performance, functionality, and reliability requirements. These also include requirements on the amount of risk that is involved in using the system.

System Reuse and Maintenance

Any useful system will not only be used for its intended application, but for other applications that may not be forseen. How flexible is the system? Can it enforce a reasonable range of security policies? If it is necessary to alter the system, can this be done without requiring a costly re-evaluation of the system's security?

4 Tradeoff Areas

In this section we discuss four major tradeoff areas that we have identified. A tradeoff area is an aspect of the system connected with security that can affect tradeoffs between security and other requirements depending on how it is handled or implemented. We have identified four: restrictions on communications, requirements on communications, security managment overhead, and interaction of a system with its environment. We discuss them in more detail below.

4.1 Restrictions on Communications

By communication we mean, not only the passing of messages between entities, but any means by which one entity can have an effect on another. Thus reading and writing are both considered communication.

Restriction on communication can be used to support integrity, guarantee of service, and confidentiality. We describe this in more detail below. We also describe some of the tradeoffs that may arise.

Restrictions on Communication in Support of Integrity

The use of restriction on communications to support integrity is straightforward: unauthorized entities are prevented from making modifications to data, and authorized entities may be only allowed to make modifications in prescribed ways. An example of the use of restrictions on communications to support integrity is that given by the Clark-Wilson model [CW87]. In this model the notion of *Constrained Data Items* is introduced. A Constrained Data Item is one that can only be operated on according to certain designated procdures. Procedures are divided into *Integrity Verification Procedures* and *Transformation Proedures*. Integrity Verification Procedures are performed to verify that the Transformation Procedures have been performed correctly. Thus the designation of Transformation Procedures may be thought of as a restriction on communication, while the designation of Integrity Verification Procedure can be thought of as a required communication.

Restrictions on Communication in Support of Guarantee of Service

A means by which restrictions on communication can be used to support guarantee of servies was described by Yu and Gligor in [YG90] and further developed by Millen [Mil92, Mil94]. In [YG90] Yu and Gligor apply the notion of *user agreements* to the denial of service problem that arises when different users compete for the same resource. An example of such a denial of service problem is the secure reader-writer problem that we discussed earlier. A user agreement describes the behavior a service use must conform to in order to be guaranteed the use of the service. If the user does not obey the agreement, then it can be denied the service. A user agreement would typically rule out behavior that would cause denial of service to others, such as the low's keeping its read lock on the data in secure reader-writer problem discussed earlier. Note that such user agreements might worsen the performance of a system in particular cases in which a user could have violated that agreement without denying the service to others, but with reducing the risk of unacceptable performance.

Restrictions on Communication in Support of Confidentiality

Restrictions on communication in support of confidentiality at first look very straightforward: an entity should not be allowed to read data it is not authorized to see. For systems that are supposed to offer an extremely high degree of data protection, such as multilevel systems which are intended to protect data classified at different security levels, the requirements can be much more stringent and present new restrictions on communication. According to the Trusted Computing System Evaluation Criteria (TCSEC) [DoD83], code must be divided into the Trusted Computing Base, which is charged with enforcing the security policy, and untrusted code. The untrusted code is usually constrained by some form of the Bell/LaPadula model [BL76], in which an untrusted process is assigned a security level and is allowed only to read data at its level and below, and only to write data at its level or above. The reason for the latter is because the untrusted process is not trusted not to copy highly sensitive data down to a less sensitive repository. This "no write-down" requirement may be further restricted for integrity reasons.

Restrictions on communications in multilevel systems become even more strict as the data protected becomes more sensitive and the risks posed leakage or disclosure become greater. At this point, it also becomes necessary to guard against the exploitation of covert channels. These are

communication channels be means of which an untrusted process running at a high security level can pass sensitive data to an untrusted process at a low security level making use of an effect the high process has on the system that is visible to the low process.

The easiest way of closing a covert channel is to partition all system resources. This of course has a very negative effect on system performance. Thus other methods have been developed, such as adding noise to a channel by making a shared resource unavailable from time to time, that can have variable effects on system performance, depending on the degree to which one wants to reduce the capacity of the channel. A number of studies documenting the tradeoffs between performance and capacity reduction have appeared in the literature. These include empirical studies that measure the results of applying various techniques on implemented channels [BCG+94], and studies that use information theory to compute the capacity of channels that may be affected by a number of different parameters [Mil87, TG88, Mos91, Gra93].

Restrictions on communication can have effects on other aspects of a system than its performance. They can also affect the availability of information. It may often happen that information is misclassified, or that it becomes necessary to prepare sanitized versions of sensitive documents. Since the sanitized documents are necessarily prepared using software that has read access to the original document, they originally exist at the higher security level, and must be downgraded. But, how we can be sure that some Trojan Horse code embedded in the software that produced the document didn't encode sensitive data in the sanitized document? For example, in [KM92] Kurak and McHugh show how to embed information in visual data in a way that is undetectable by human review. Thus it is clear that, although it may sometimes by necessary, downgrading should be treated very cautiously, and the tradeoff between confidentiality and availability of data should always be kept in mind.

A third way in which restrictions on communication can affect a system is in the integrity and consistency of the data. For example, consider the oft-cited example of the plane with two types of cargo, one classified and one unclassified, and a limit on how much weight the plane can carry [SD86]. If the user entering the unclassified cargo is told exactly how much weight it available, and if the total limit is unclassified, then he or she can tell exactly what the weight of the classified cargo is. Even if the total limit is not revealed, the available weight parameter still has the potential of being used as a covert channel. It is possible to close this channel and still maintain integrity by allowing fixed maximum weights for both classified and unclassified cargo, but this has the possible effect of preventing the aircraft from being used to its maximum capacity. Thus there is a three-way tradeoff between integrity, confidentiality, and performance.

4.2 Required Communications

If a system is distributed, so that some communication between components takes place in a hostile environment that may include passive or active eavesdroppers, than additional communications may be required in order to authenticate various components of the system to each other and to distribute cryptographic keys so that components can communicate securely. Traditional low-cost solutions such as passwords do not provide sufficient protection in such an environment; an intruder can simply read the password as it goes across the communication channel and use it to gain access to the system. In order to gain even a reasonable degree of security, some use of encryption, both for secrecy and authentication, is required. This means that protocols for key distribution and authentication need to be introduced that can increase the burden on a system in a number of ways.

In general, there are three ways in which the introduction of cryptographic protocols for authentication and secrecy can affect system performance. One of these is the number or length of

messages. A mutual authentication protocol involves at least three messages, for example. A key distribution protocol will require five or six, or more depending upon the application involved. The second added cost is the expense of performing the encryption operation itself. The third added cost is the overhead required to manage, guard, and generate cryptographic keys and authentication information. A cryptographic authentication system must make the use of one or more authentication servers whose job it is to provide the information that parties using the system need to authenticate each other, as well possible to generate keys that will be used for secure communication.

Tradeoffs between security and performance in this area are not necessarily clear-cut, but in general there is some correlation between the number of messages used in an authentication protocol and the degree of security that is achieved. In [Gon93] a study of several kinds of protocols describing the degree of security of each kind together with the number of messages needed to achieve it makes this tradeoff explicit.

It is possible to trade off the various performance costs of secure communication against each other in some cases. The use of public key cryptography can greatly reduce the amount of management overhead needed. When single key cryptography is used, the authentication server must know a master key for each principal, which must be kept secret. Any response to a request to have a session key delivered to a principal must be responded to by a message encrypted with that principal's master key. On the other hand, if public key cryptography is used, only the principals' public keys, which do not need to be kept secret, need to be known. It is possible even for the authentication server to go "off-line" by providing each principal a certificate signed with the server's private key that contains that principal's name and public key as well as any other relevant information. However, public-key cryptography is usually more computationally expensive than single-key, enough so that performance can be visibly effected by its use. Thus in this case we are trading off performance costs against key management overhead costs.

There are other required communications besides those of authentication. All secure systems require keeping some sort of audit record, so that security violations can be detected and repaired. This requires that system activities be recorded in an audit log, thus presenting the system designer with a set of required communications. Also, when access control is used, requests for accesses must be presented to a reference monitor that decides whether or not the access can be granted, giving another example of a required communication. Usually, these added communications do not affect system performance, except, when they take in the context of a distributed system, in which case, not only do the same authentication tradeoffs that we have discussed above arise, but so do some others connected with maintaining consistency and timeliness of audit and access data. We will discuss this in more detail in the next section.

4.3 Security Management Overhead

So far we have restricted ourselves to the study of security as it affects communication within a system. In this section we consider the cost of the overhead that arises from the security mechanisms themselves. There are a number of ways in which security management overhead can affect a system. These include the performance impact of access checking, the performance impact of audit trail analysis, the time and cost of developing and evaluating the security mechanisms, and the degree of difficulty of modifying the system without affecting the security mechanisms. We discuss all these below.

Performance Impact of Audit Trail Analysis

An audit trail contains information that can be vital to the security of a system. Assuming that the security of auditing mechanism has not been breached, the audit trail will contain information that will be useful in identifying security violations and attempted security violations. The problem, of course, is in finding this information in a huge audit trail consisting largely of innocuous data. Although intrusion detection by audit trail analysis is an active field of research, this is a problem that is still unsolved. Thus one must take into account the tradeoffs between the amount of information that can be gained from audit trail analysis and the negative performance impact a more thourough analysis will have.

Performance Impact of Access Checking

In a secure system, each access control must be checked and only allowed if it is authorized. In cases in which the entity performing the access control check is on the same machine as the data or programs being accessed, performance does not seem to be greatly affected. But, if the access checker and the data are on different machines, it will probably be necessary to use cryptography and authentication protocols to ensure the identity of the entity requesting the checks, the entity performing the checks, and the entity that is being accesses. This can have a negative effect on system performance.

There are several ways in which tradeoffs can be introduced to increase performance in this case. One is to use public-key cryptography and have the authorizing entity produce certificates signed with its private key that give other entities various rights. These decrease communication costs, but at the cost of making it difficult to revoke access. This can be in part be offset by having the certificates expire and reissued periodically, thus introducing a tradeoff between the inconvenience of reissuing certificates and the possible threat to security of being unable to revoke them in time.

Another way to help performance is to distribute access control authority so that access checking could be done at a number of points in the system. There are several ways this could be implemented. One would be to allow more than one access control point to grant or deny access to the same entity. This gives the greatest availability, but at the cost of possible inconsistency between access control points. Another solution is to grant different domains to each access control point. These access control points are in turn governed by access control points that mediate between domains. This may be done several times, so that a hierarchy is formed. This improves performance within a domain, but has the potential of worsening it between domains, since the hierarchy must be traversed whenever such an access is granted.

System Development and Evaluation

In the previous sections we talked about the effect security management overhead can have on performance. But it can have an even greater effect on system development and evaluation. System development and evaluation time is the greatest hidden cost of a secure system. In order for a secure system to be usable as such, it must not only be secure, it must be believed to be secure. But often the work required to provide such assurance can add a significant to the development time and expense of a system. The extended time necessary for evaluation can have a hidden negative effect on system performance; often system performance suffers not as a direct result of implementing security mechanisms but because the system evaluation takes such a long time that the latest means for improving system performance are not used, simply because they were not available at the time the system was built.

There is often a direct tradeoff between complexity of assurance and restrictions on communication. Earlier we noted that the TCSEC mandates the division of a system into a Trusted Computing Base which is charged with enforcing the security policy, and untrusted code which is constrained by the security policy. This was somewhat of an oversimplification: the TCB contains more than the part of the system charged with enforcing the policy. It also contains code that is not constrained by the policy (for example, the restrictions on communication given by the Bell-LaPadula model). Thus, code in the TCB may have the ability to read and write at all security levels. However, it is "trusted" not to cause any harm, that is, not to leak data from one security level to a lower or incomparable one. This trust is a result of careful design and evaluation. Thus, introducing trusted code will increase system development and evaluation time, but it may also improve performance by removing restrictions on necessary functions.

There are several strategies for reducing the amount of labor involved in providing assurance. One, of course, is to keep the size of the part of the system charged with enforcing the security policy as small and well-structured as possible. This is a well-known principle, that is used as the basis of the TCSEC. Another, perhaps less well-known strategy, is to limit the kinds of secure systems that can be developed. The reasoning behind this strategy is that, if only certain kinds of systems can be developed, then the techniques for securing them and assuring that the security is adequate will become well enough understood so that they can be applied in a timely fashion. This, for example, was the strategy behind the TCSEC, to only allow a small number of classes of secure systems that could however be used in a wide variety of applications [Pot94]. This approach of course has the danger that the limits may be so severe that it is not possible to build compliant systems that enforce necessary security requirements. This indeed is a complaint that is often made against the TCSEC; thus the newer criteria, such as the European ITSEC [ITS91] generally provide more flexibility for the designer. However, this greater flexibility has the potential of increasing the amount of work needed to provide assurance; decisions that were hardwired in the more restricted approach now must be justified.

Another approach, recommended by Sterne et al. in [SBT94] is to "spread out" assurance so that multilevel security can be enforced by compenents that offer different degrees of assurance. Sterne et al. propose the use of *Controlled Application Sets* that are intended to be trusted to not to the same extent as a TCB, and whose behavior is constrained by the TCB. One of the purposes of a CAS would be to contain programs that are reasonably certain to not to contain hostile Trojan Horse code that would attempt to exploit covert channels to leak sensitive information. Thus a member of a CAS could be safely allowed to access sensitive information even in a system which is known to have covert channels. In this case the proposes technique has not seen enough use for us to determine what the exact tradeoffs would be, but at the very least there appears to be a tradeoff between the amount of code that would have to be assured, and the degree of work that would need to go into assuring the individual pieces.

System Reuse and Modification

One of the biggest obstacles to secure system reuse is the difficulty of modifying security mechanisms without having to undergo a lengthy re-evaluation. For example, introducing new trusted code into the TCB generally requires a complete re-evaluation of the TCB. This is to be avoided at all costs. One method for avoiding this problem is to use a layered TCB approach, or TCB subsetting [SS87]. In such an approach additional security functionality is built on top of the original TCB and is considered "untrusted" by the original TCB. Thus the original TCB is not modified and does not have to be re-evaluated. An example would be a multilevel secure database management system in which mandatory access control (MAC) protecting data classified at different security levels is

supplies by the underlying operating system, while discretionary access control (DAC) is provided by the by database management system itself. This is necessary since the DAC offered by the operating system protects only files, not tuples or relations and thus is not fine-grained enough for the purposes of a DBMS. Guidelines for implementing TCB subsetting are given in the Trusted Database Interpretation of the TCSEC [Nat91]. We note that Sterne et al.'s Controlled Application Sets discussed in the previous section also have much of this flavor.

There is a tradeoff involved in this approach, however. In general, although not always, the degree of assurance offered by the higher levels of the TCB is less than that offered by the lower layers. In the case of the database management system example, the assurance of the DBMS's DAC is not as high as the assurance of the TCB's DAC. Thus, this approach is usually recommended for cases in which different security goals require different degrees of assurance.

4.4 Environment

The environment in which a system will operate can have a major impact on its security requirements. A low-risk environment will require a less secure system, and thus reduce many of the costs we have discussed above. It is also possible to reduce risk, not by changing the environment, but by putting restrictions on the way the system interacts with the environment, that is, by decreasing the functionality of a system. To give some examples, consider an ATM system which users can only use to find out the size of their bank accounts versus one in which users can also withdraw cash and transfer money from one account to another [LL85], or consider a system which contains data classified at only one security level versus a system that protects data classified at different security levels.

The degree to which risk can be traded off against functionality and environmental restrictions is not that well understood, but in some special cases rules of thumb have been worked out. In general, it is understood that, the less functionality that a system offers, and the more restricted the environment, the less security functionality and assurance is required, This is the idea behind much of the work on risk analysis for security, and it has also been codified for use in deciding what TCSEC rating is necessary for a system. For example DoD Directive 5200.28 [DoD88] provides guidelines for the TCSEC rating required based on the range of security levels of data, and the range of user clearances; the greater the range of security classifications or clearances, the higher the TCSEC rating required. In other words, the greater the security functionality required, the greater the degree of assurance required. The work of Landwehr and Lubbes [LL85] takes this approach even further by including risk factors such as the local processing capability available to the user (e.g., programmable terminals versus fixed-function interactive terminals), the type of communication paths, and the capabilities the system gives to the user (e.g., transaction processing versus full programming). All of these risk factors involve adding more functionality to a system. Thus in this case the tradeoffs are between functionality and risk. When risk is increased the increased cost appears in the necessity for greater system assurance.

5 Conclusion

In this paper we have looked at the tradeoffs invoved in secure system development from three points of view: types of security requirements, points in the life cycle of a system, and areas in which tradeoffs are most likely to occur. We have given some examples of some of the more common tradeoffs, and we have shown how, when investigating tradeoffs, it is important to have the entire life cycle of a system in mind.

References

[BCG+94] P. K. Boucher, R. K. Clark, I. B. Greenberg, E. D. Jensen, and D. M. Wells. Toward a Multilevel-Seucre, Best-Effort Real-Time Scheduler. In *Proceedings of DCCA4*, pages 33–45. January 1994.

[BL76] D. E. Bell and L. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report MTR-2997, MITRE Corporation, March 1976. Available as NTIS AD A023 588.

[CW87] D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Security Policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 184–194, April 27-29 1987.

[DoD83] Department of Defense Trusted System Evaluation Criteria. Technical Report CSC-STD-001-83, Department of Defense Computer Security Center, August 15 1983.

[DoD88] Security Requirements for Automated Information Systems. DoD Directive 5200.28, March 21 1988.

[Gon93] Li Gong. Lower Bounds on Messages and Rounds for Network Authentication Protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 26–37. Association for Computing Machinery, November 1993.

[Gra93] J. W. Gray. On Analyzing the Bus-Contention Channel under Fuzzy Time. In *Proceedings of the Computer Security Workshop VI*. IEEE Computer Society Press, June 1993.

[ITS91] Information Technology Security Evaluation Criteria (ITSEC): Provisional Harmonized Criteria. Document COM(90)314, Office for the Official Publications of the European Communities, June 1991.

[KM92] C. Kurak and J. McHugh. A Cautionary Note on Image Downgrading. In *Proceedings of the Eighth Annual Computer Security Applications Conference*, pages 153–159. IEEE Computer Society Press, Los Alamitos, California, 1992.

[LL85] Carl E. Landwehr and H. O. Lubbes. An Approach to Determining Computer Security Requirements for Navy Systems. NRL Report 8897, Naval Research Laboratory, May 13 1985.

[Mil87] J. K. Millen. Covert Channel Capacity. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, April 1987.

[Mil92] J. K. Millen. A Resource Allocation Model for Denial of Service. In *1992 Symposium on Research in Security and Privacy*, pages 137–147. IEEE Computer Society Press, May 1992.

[Mil94] J. K. Millen. Denial of Service: A Perspective. In *Proceedings of DCCA4*, 1994. to appear.

[Mos91] Ira Moskowitz. Variable Noise Effects Upon a Simple Timing Channel. In *Proceedings of the 1991 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 1991.

[Nat91] National Computer Security Center. *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*, April 1991.

[Pot94] Garrell Pottinger. Proof Requirements in the Orange Book: Origins, Implementation, and Implications. Technical report, Mathematical Sciences Institute, Cornell University, Feb. 11 1994.

[SBT94] D. F. Sterne, G. S. Benson, and H. Tajali. Redrawing the Security Perimeter of a Trusted System. In *Proceedings of the 7th Computer Security Foundations Workshop*, pages 162–174. IEEE Computer Society Press, June 14-16 1994.

[SD86] R. R. Schell and D. E. Denning. Integrity in trusted database systems. In *Proceedings of the National Computer Security Conference*, pages 30–36. NCSC, September 15-18 1986.

[SS87] W. R. Shockley and R. R. Schell. TCB Subsets for Incremental Evaluation. In *Proceedings of the Third Aerospace Computer Security Conference*, pages 131–139, Orlando, FL, December 1987.

[TG88] C.-R. Tsai and V. D. Gligor. A Bandwidth Computation Model for Covert Storing Channels and its Application. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, April 1988.

[YG90] C.-F. Yu and V. D. Gligor. A Specification and Verification Method for Preventing Denial of Service. *IEEE Transactions on Software Engineering*, 16(6):581–592, June 1990.